REMARKS/ARGUMENTS

Claims 1-26 are pending in the above-captioned application. An Office Action dated July 14, 2003 (the Office Action) rejected claims 1-26. Claims 2, 4, 19, 21, 23 and 25 have been cancelled and claims 1, 3, 5, 6, 18, 20, 22, 24 and 26 have been amended.

Rejections under 35 U.S.C. §103

The Office Action rejected claims 1-9 and 14-26 as unpatentable over United States Patent No. 6,317,869 to Adl-Tabatabai in (Adl-Tabatabai) view of United States Patent 5,692,193 to Jagganathan (Jagganathan).

Claim 1 as amended recites a method of managing memory in a multi-threaded processing environment including local thread stacks and local thread heaps, and a global heap, said method comprising:

creating an object in a thread heap;

for a given thread, monitoring each object in its heap, to determine whether the object is accessible by any thread other than the given thread;

assigning a status to the given object, the status designating the object as a local object; and

changing the status of the object to global when the monitoring step determines that the object is accessible from either of a global root or other global object.

The Office Action concedes that Adl-Tabatabai does not disclose monitoring each object in the heap. However, the Office Action contends that Jagganathan teaches this element and that it would have been obvious to one skilled in the art to modify Adl-Tabatabai according to Jagganathan. Applicant respectfully submits that the claims at issue would not have been obvious in view of the combination of Adl-Tabatabai and Jagganathan.

The Office Action contends that Adl-Tabatabai discloses the invention as claimed except that it does not disclose the second element of claim 1 (determining whether the object is accessible by

7

any thread other than the given thread). Specifically, the OA contends that the first element is taught by Adl-Tabatai at col. 5, lines 13-15 which read as follows:

"However, reference values do not point to primitives or non-reference values. An object is created in the Java heap, and is garbage collected after there are no more references to it."

The OA further contends that the second element of claim 1 is disclosed by Adl-Tabatai at col. 12, line 65- col. 13, line 27, which reads:

"Sting provides thread groups as a means of gaining control over a related collection of threads. A thread group is created by a call to fork-thread-group; this operation creates a new group and a new thread that becomes the root thread of that group. A child thread shares the same group as its parent unless it explicitly creates a new group. A thread group includes a shared heap accessible to all its members. When a thread group terminates via the call, (thread-group-terminate group), all live threads in the group are terminated and its shared heap is garbage collected.

A thread group also contains debugging and thread operations that may be applied en masse to all of its members. Thread groups provide operations analogous to ordinary thread operations. (e.g., termination, suspension, etc.) as well as operations for debugging and monitoring (e.g., listing all threads in a given group, listing all groups, profiling, genealogy information, etc..) Thus, when thread T is terminated, users can request all of T's children (which are defined to be part of T's group to be terminated) thus: (thread-group-terminate (thread.group T))

Thread groups are an important tool for controlling sharing in a hierarchical memory architecture. Since objects shared by members in a groups are contained in the group's shared heap, they are preferably physically close to one another in memory, and thus better locality is exhibited."

8

Sting is a dialect of Scheme (described by Jonathan Rees and William Clinger, editors in "The Revised Report on the Algorithmic Language Scheme" in ACM Sigplan Notices, 21(12), 1986) comprising an operating system which is a feature of a preferred embodiment of the Jagganathan invention.

To establish obviousness the combination of references must not only disclose each element of a claim but must also provide a motivation or reason to combine the references. In this case, the Office Action contends that the motivation to combine is as follows:

"It would have been obvious to a person skill [skilled] in the art at the time the invention was made to combine the teaching of Adl-Tabatabai and Jagganathan because Jagganathan's teaching of monitoring object[s] in a heap would allow each thread to perform garbage collection independently of one another."

Adl-Tabatabai relates to a method of run-time tracking of object references that uses a bit vector for an ambiguously typed variable and maintaining a bit for determining whether the variable is assigned a value. See claim 1. The Adl-Tabatabai appears to be a solution of the problem of mistakenly considering a value as a reference value in a garbage collection process. That has nothing to do with the claimed invention which relates to assigning of status for objects as either local or global.

Jagganathan neither teaches nor suggests "for a given thread, monitoring each object in its heap, to determine whether the object is accessible by any thread other than the given thread" and does not provide any motivation for combining the references. Certainly, the Office Action rationale for making the combination is neither taught nor suggested by Jagganathan or the combination of the cited references.

There is no mechanism in Jagganathan for tracking the locality of objects; Jagganathan specifically states the rules which govern the placement of objects in the stack, thread local heap

9

and global heap. The placement decision is made once and not subsequently altered. The claimed mechanism by contrast specifically teaches the mechanism by which an object can be assigned to a local heap and then moved to the global heap when it becomes globally reachable.

Therefore, those skilled in the art would not have been motivated by either of the cited references, whether viewed individually or in combination, to make the claimed invention.

Applicants provide the following explanation to aid in understanding the claimed invention. However, it is not intended to limit the scope of the claims. Thread private or local heaps are used to allocated non-shared objects whose lifetimes might exceed the lifetime of the procedures that created them. The term "might exceed" is used because it is not always possible for the compiler to determine the lifetime of an object in programming languages such as Scheme or ML. Furthermore, it may not be possible to determine the lifetimes of objects in languages which allow calls to unknown procedures. References contained in private heap can refer to other objects in the same private heap, or objects in shared or global heaps, but they cannot refer to objects in the stack. References in the stack may refer to objects in the private heap, but references in the shared heap may not. Private heaps lead to greater locality since data allocated on them are used exclusively by a single thread of control; the absence of interleaving allocations among multiple threads means that objects close together in the heap are likely to be logically related to one another.

No other thread can access objects that are contained in a thread's stack or local heap. Thus, both thread stacks and local heaps can be implemented in local memory on the processor without any concern for synchronization or memory coherency. Thread local heaps are actually a series of heaps organized in a generational manner. Storage allocation is always done in the youngest generation in a manner similar to other generational collectors. As objects age they are moved to older generations. All garbage collection of the local heap is done by the thread itself. In most thread systems that support garbage collection all threads in the system must be suspended during a garbage collection. In contrast, Sting's threads garbage collect their local

10

heaps independently and asynchronously with respect to other threads. Thus other threads can continue their computation while any particular thread collects its local heap; this leads to better load balancing and higher throughput. A second advantage of this garbage collection strategy is that the cost of garbage collecting a local heap is charged only to the thread that allocates the storage, rather than to all threads in the system.

The Office Action contends that the limitation of claim 3 (deleting from the thread heap one or more local objects when it is determined that they are not accessible from a local root) is disclosed at Adl-Tabatabai col. 7, lines 29-46. That section relates to garbage collection and does not suggest the claimed combination. Moreover, claim 3 is distinguishable over the cited references for the same reasons that claim 1 is patentable.

The Office Action contends that the limitation of claim 5 (changing the status of an object in the thread heap to global if the object is assigned to a static variable or if the object is assigned to a field in a global object) is disclosed at Adl-Tabatabai col. 5, lines 29-30 and 43-44. As discussed, above Adl-Tabatabai neither teaches nor suggests changing the local/global status of an object. Moreover, claim 5 is distinguishable over the cited references for the same reasons that claim 1 is patentable.

The Office Action contends that the limitation of claim 6 (changing the status of an object in the thread heap to global if the object is assigned to a static variable or if the object is assigned to a field in a global object) is disclosed at Adl-Tabatabai item 540 in fig. 5B. Item 540 is the mere assignment of a value to a variable at the beginning of a method; it does not relate to changing the status. As discussed, above Adl-Tabatabai neither teaches nor suggests changing the local/global status of an object. Moreover, claim 6 is distinguishable over the cited references for the same reasons that claim 1 is patentable.

Claims 7-9, 14, 15, 17, 18-26 are distinguishable over the cited combination for reasons discussed above.

11

The Office Action contends that the limitation of claim 16 (analysing whether an object is likely to be made global and associating such an object with a global status on creation) is disclosed at Adl-Tabatabai col. 4, line 58 – col. 5, line 9). That section says nothing about the *likelihood* of an object being global. Moreover, claim 16 is distinguishable over the cited references for the same reasons that claim 1 is patentable.

The Office Action rejected claims 10-13 as unpatentable over United States Patent No. 6,317,869 to Adl-Tabatabai in (Adl-Tabatabai) view of United States Patent 5,692,193 to Jagganathan (Jagganathan) and further in view of United States Patent 6,308,315 to Dice.

The Office Action admits that the limitation of claim 10 (using spare capacity in an object header for the status) is not disclosed by either Adl-Tabatabai or Jagganathan but that the subject matter of claim 10 would have been obvious in view of Dice col. 6, lines 35-38. That section discusses object headers having space for variables – a well known general fact – but says nothing about using spare capacity for status. In addition the rationale used in the Office Action for combining three disparate references was that the combination would make garbage collection faster and more precise. That is an inadequate and unsupported (by any evidence of record) rationale for combining the references. First the statement is so general that if that were sufficient then any combination of references could be combined thus rendering a entire class of valuable inventions obvious. That cannot be the standard for obviousness. Moreover, claim 10 is distinguishable over the cited references for the same reasons that claim 1 is patentable.
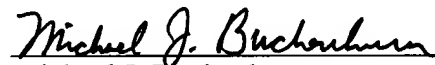
The Office Action contends that the limitation of claim 11 (using multiples of 2 or more bytes in a thread heap to store the objects) is disclosed at Adl-Tabatabai col. 5, lines 5-7. However, the Office Action does not even allege that the limitation "whereby there is at least one spare bit in the object length variable and using the at least one spare bit as the status" is disclosed anywhere in the cited references. Moreover, claim 11 is distinguishable over the cited references for the same reasons that claim 1 is patentable.

12

The Office Action contends that the limitation of claim 12 (moving objects whose status is global from the thread heap to the global heap) is disclosed at Jagganathan col. 14, lines 39-42. Applicant respectfully traverses this statement. Jagganathan neither teaches nor suggests this limitation. The cited portion discusses moving half of the TCBs in the local pool to the global pool when the local pool overflows. That has nothing to do with the claimed limitation.

The Office Action contends that the limitation of claim 13 (compacting the reachable local objects in a thread heap) is disclosed at Adl-Tabatabai col. 2, lines 30-32. Applicant respectfully traverses this statement. That section discusses compacting memory space by moving objects it finds to another region. It says nothing about the thread heap.

For the foregoing reasons, Applicant respectfully requests allowance of the pending claims and that a timely Notice of Allowance be issued in this case.

Respectfully submitted,

Michael J. Buchenhorner

Michael J. Buchenhorner
Reg. No. 33,162

Date: November 14, 2003

HOLLAND & KNIGHT LLP
Holland & Knight LLP
701 Brickell Avenue, Suite 3000
Miami, FL 33131
(305) 789-7773 (voice)
(305) 789-7799 (fax)

13

Certificate of First Class Mailing

I hereby certify that this Amendment and Response to Office Action is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 on November 14, 2003.

Michael J. Buchenhorner
Michael J. Buchenhorner

Date: November 14, 2003

1358294_v2